Lab: Objects, Classes and Collections

This document defines the exercises for "Java Advanced" course @ Software University. Please submit your solutions (source code) of all below described problems in Judge.

I. Stacks

1. Matching Brackets

We are given an **arithmetical expression** with **brackets**. Scan through the string and extract each **sub-expression**.

Print the each sub-expression on a new line.

Examples

Input	Output
1 + (2 - (2 + 3) * 4 / (3 + 1)) * 5	(2 + 3) (3 + 1) (2 - (2 + 3) * 4 / (3 + 1))
(2 + 3) - (2 + 3)	(2 + 3) (2 + 3)

Hints

- Use a stack, namely an **`ArrayDeque()`**
- Scan through the expression searching for brackets
  - If you find an opening bracket, push the index into the stack
  - If you find a closing bracket pop the topmost element from the stack. This is the index of the opening bracket.
  - Use the current and the popped index to extract the sub-expression

```
if (ch == '(') {
    stack.push(index);
}
else if (ch == ')') {
    int startIndex = stack.pop();
    String contents = expression.substring(startIndex, index + 1);
    System.out.println(contents);
}
```

II. Queues

2. Hot Potato

Hot potato is a game in which children in a circle start passing a hot potato. The counting starts with the fist kid. **Every nth time the child left with the potato leaves the game**. When a kid leaves the game, it passes the potato **forward**. This continues repeating **until there is only one kid left**.

Create a program that simulates the game of Hot Potato. **Print every kid that is removed from the circle**. In the end, **print the kid that is left last**.

On the first line you will get the **children names** and on the second – the integer **n**.

Examples

Input	Output
Mimi Pepi Toshko 2	Removed Pepi Removed Mimi Last is Toshko
Gosho Pesho Misho Stefan Krasi 10	Removed Krasi Removed Pesho Removed Misho Removed Gosho Last is Stefan
Gosho Pesho Misho Stefan Krasi 1	Removed Gosho Removed Pesho Removed Misho Removed Stefan Last is Krasi

3. Math Potato

Rework the previous problem so that a **child is removed only on a prime cycle** (cycles start from 1)

If a **cycle is not prime**, just **print the child's name.**

As before, print the name of the child that is left last.

Examples

Input	Output
Mimi Pepi Toshko 2	Removed Pepi Prime Mimi Prime Toshko Removed Mimi Last is Toshko
Gosho Pesho Misho Stefan Krasi 10	Removed Krasi Prime Pesho Prime Misho Removed Stefan Prime Gosho Removed Gosho Prime Misho Removed Pesho Last is Misho

III. Sets

4. Parking Lot

Write program that:

- Records a **car number** for every car that enters a **parking lot**
- Removes a **car number** when a car goes out

The input will be a string in format **`[direction, carNumber]`** and ends with the string **`"END"`**

Print the all car numbers which are in the parking lot at the end.

The order of the output does not matter.

Examples

Input	Output
IN, CA2844AA IN, CA1234TA OUT, CA2844AA IN, CA9999TT IN, CA2866HI OUT, CA1234TA IN, CA2844AA OUT, CA2866HI IN, CA9876HH IN, CA2822UU END	CA2822UU CA2844AA CA9999TT CA9876HH
IN, CA2844AA IN, CA1234TA OUT, CA2844AA OUT, CA1234TA END	`Parking Lot is Empty`

Hints

- Car numbers are **unique**
- Use the methods **`isEmpty()`**

Solution

You might help yourself with the code below:

```
while(true) {
    String input = sc.nextLine();
    if (input.equals("END")) {
        break;
    } else {
        String[] reminder = input.split( regex: ", ");
        if (reminder[0].equals("IN")) {
            parkingLot.add(reminder[1]);
        } else {
            parkingLot.remove(reminder[1]);
        }
    }
}
```

5. SoftUni Party

There is a party in SoftUni. Many guests are invited and they are of two types - **VIP** and **regular**. You will get a reservation list with all invited guests. When a guest comes, check if he/she exists in the reservation list. **All** reservation numbers will consist of **8 chars**. All **VIP** numbers **start with a digit**.

There will be 2 command lines:

- "PARTY" – the party is on and guests start coming.
- "END" – the party is over and no more guest will come.

Output

- On the first line, print how many guests didn't show up
- On the lines after that print their reservation numbers
- First, print all VIP guests in lexicographic order and after that print the regular guests, again in lexicographic order.

Examples

Input	Output	Input	Output
7IK9Yo0h 9NoBUajQ Ce8vwPmE SVQXQCbc tSzE5t0p PARTY 9NoBUajQ Ce8vwPmE SVQXQCbc END	2 7IK9Yo0h tSzE5t0p	2FQZT3uC dziNz78I fPXNHpm1 HTTbwRmM xys2FYzn MDzcM9ZK PARTY 2FQZT3uC dziNz78I fPXNHpm1 HTTbwRmM END	2 MDzcM9ZK xys2FYzn

IV. Maps

6. Academy Graduation

Write a program that reads students with their **names** and **scores** for different courses. Afterwards print the average score of every student in the following format: "**{Students name}** is graduated with **{average score)**"

The order of the output does not matter.

Examples

Input	Output
3 Gosho 3.75 5 Mara 4.25 6 Pesho 6 4.5	Gosho is graduated with 4.375 Mara is graduated with 5.125 Pesho is graduated with 5.25
5 Gruio 4.36 5.50 3.30 5.63 2.57 5.75 2.81 4.89 Trendafilka 3.10 5.35 3.30 3.35 5.64 4.99 2.75 4.68 Mite 3.45 3.23 3.03 5.42 5.46 4.15 2.26 5.95 Roza 2.08 3.48 3.36 2.73 2.96 4.54 3.70 3.85	Ganio is graduated with 4.09375 Gruio is graduated with 4.351249999999999 Mite is graduated with 4.11875 Roza is graduated with 3.3375 Trendafilka is graduated with 4.145

Ganio 4.75 4.92 3.78 4.79 4.82 4.75 2.81 2.13	

Hints

- Think about a **proper type** of map
- **Values** can be stored into an **array**